Radio Button

The <aava-radio-button> component provides a radio button group for single-choice selections with multiple layout orientations, size options, custom color theming, and smooth interaction effects. It includes full Angular forms integration and accessibility compliance.

How to use

Basic Usage

Simple radio button group implementation with multiple options and two-way data binding.

```
import { AavaRadioButtonComponent } from "@aava/play-core";
```

Orientations

The radio button component supports both horizontal and vertical layouts to accommodate different design requirements.

```
&lt;aava-radio-button
  [options]="[
                    { label: 'Option 1', value: 'option1' },
                    { label: 'Option 2', value: 'option2' },
                    { label: 'Option 3', value: 'option3' },
                  ]"
  name="basic-radio"
  selectedValue="option1"
  (selectedValueChange)="onRadioChange($event)"
&gt;
&lt;/aava-radio-button&gt;
```

Available Orientations

- vertical - Stacked layout with options arranged vertically (default)
- horizontal - Inline layout with options arranged horizontally

Sizes

Three size options to match different interface densities and visual hierarchies.

```
  onRadioChange(value: string) {
    console.log('Radio button selection changed:', value);
  }
```

Available Sizes

- sm (Small) - Compact size for dense interfaces
- md (Medium) - Standard size for most use cases (default)
- lg (Large) - Prominent size for important selections

States

Various states for different user scenarios including selection, disabled, and interactive states.

```
&lt;aava-radio-button
  [options]="[
                  { label: 'Option 1', value: 'option1' },
                  { label: 'Option 2', value: 'option2' },
                  { label: 'Option 3', value: 'option3' },
                ]"
  name="vertical-radio"
  orientation="vertical"
  selectedValue="option2"
  (selectedValueChange)="onRadioChange('vertical', $event)"
&gt;
&lt;/aava-radio-button&gt;
&lt;aava-radio-button
  [options]="[
                  { label: 'Option 1', value: 'option1' },
                  { label: 'Option 2', value: 'option2' },
                  { label: 'Option 3', value: 'option3' },
                ]"
  name="horizontal-radio"
  orientation="horizontal"
  selectedValue="option3"
  (selectedValueChange)="onRadioChange('horizontal', $event)"
&gt;
&lt;/aava-radio-button&gt;
```

Available States

- Unselected - Default state for non-selected options
- Selected - Active state showing the selected option
- Disabled - Non-interactive state for unavailable options

Custom Colors

Comprehensive color customization for radio buttons and labels to match design requirements.

```
  onRadioChange(orientation: string, value: string) {
    console.log(`${orientation} orientation selection changed:`, value);
  }
```

Color Customization Options

- labelColor - Custom color for option labels
- color - Cutom color for the dot

Form Integration

Form integration with ControlValueAccessor implementation for reactive from

```
&lt;aava-radio-button
  [options]="[
          { label: 'Option 1', value: 'option1' },
          { label: 'Option 2', value: 'option2' },
          { label: 'Option 3', value: 'option3' }
        ]"
  name="small-radio"
  size="sm"
  selectedValue="option1"
&gt;
&lt;/aava-radio-button&gt;
&lt;aava-radio-button
  [options]="[
          { label: 'Option 1', value: 'option1' },
          { label: 'Option 2', value: 'option2' },
          { label: 'Option 3', value: 'option3' }
        ]"
  name="medium-radio"
  size="md"
  selectedValue="option2"
&gt;
&lt;/aava-radio-button&gt;
&lt;aava-radio-button
  [options]="[
          { label: 'Option 1', value: 'option1' },
          { label: 'Option 2', value: 'option2' },
          { label: 'Option 3', value: 'option3' }
        ]"
  name="large-radio"
  size="lg"
  selectedValue="option3"
&gt;
&lt;/aava-radio-button&gt;
```

Reactive Form Features

- FormControl binding - Direct integration with Angular reactive forms
- Value tracking - Proper change detection and emission
- Disabled control - Programmatic enable/disable support

Accessibility Support

The Radio Button component provides comprehensive accessibility features to ensure compatibility with screen readers, keyboard navigation, and mobile devices.

ARIA Support

- role="radiogroup" - Container identifies as a radio button group
- role="radio" - Each option is properly identified as a radio button
- aria-checked - Indicates selection state (true/false)
- aria-disabled - Identifies disabled options
- aria-label - Describes the radio group ("Radio button group with X options")
- aria-labelledby - Links radio buttons to their text labels
- aria-describedby - Provides additional context for screen readers
- aria-hidden="true" - Hides decorative elements from assistive technology

Keyboard Navigation

- Arrow Keys (↑↓←→) - Navigate between radio options
- Space/Enter - Select the currently focused option
- Tab - Enter and exit the radio group
- Circular Navigation - Arrow keys wrap around when reaching first/last option
- Skip Disabled - Navigation automatically skips disabled options
- Visual Focus - Clear focus indicators show current keyboard position

WCAG 2.1 Compliance

- Level AA compliant for color contrast
- Keyboard Accessible - All functionality available via keyboard
- Focus Management - Logical focus order and clear indicators
- Semantic Structure - Proper HTML roles and relationships
- Error Prevention - Clear state indication prevents user confusion

API Reference

Inputs

Property	Type	Default	Description
options	RadioOption[]	[]	Array of radio button options
name	string	''	HTML name attribute for the radio group
selectedValue	string	''	Currently selected option value
size	'sm' | 'md' | 'lg'	'md'	Size of radio buttons
orientation	'horizontal' | 'vertical'	'vertical'	Layout orientation
color	string	''	Custom color for radio button and glow
labelColor	string	''	Custom color for option labels
radio	'dot' | 'none'	'dot'	Whether to show the inner dot
animation	'none' | 'shadow'	'none'	Animation effect for selection

Outputs

Event	Type	Description
selectedValueChange	EventEmitter<string>	Emitted when selection changes

CSS Custom Properties

Property	Description
--radio-group-gap	Gap between radio button options

Property	Description
--radio-checkmark-background	Background color of the radio button circle
--radio-checkmark-border	Border color of the radio button
--radio-checkmark-border-radius	Border radius for the radio button
--radio-checkmark-background-disabled	Background when disabled
--radio-checkmark-border-disabled	Border color when disabled
--radio-dot-background	Background color of the inner dot
--radio-dot-border-radius	Border radius of the inner dot
--radio-dot-background-disabled	Dot background when disabled
--radio-label-color	Text color for option labels
--radio-label-font	Font properties for labels
--radio-label-margin-left	Left margin for labels
--radio-label-color-disabled	Label text color when disabled
--radio-label-cursor-disabled	Cursor style when disabled
--radio-cursor	Cursor style for interactive elements
--radio-cursor-disabled	Cursor style when disabled
--radio-custom-glow-color	Custom glow color for selected state
--radio-size-sm	Size dimensions for small variant
--radio-size-md	Size dimensions for medium variant
--radio-size-lg	Size dimensions for large variant
--radio-size-sm-dot	Dot size for small variant
--radio-size-md-dot	Dot size for medium variant
--radio-size-lg-dot	Dot size for large variant
--radio-size-sm-label	Label font size for small variant
--radio-size-md-label	Label font size for medium variant
--radio-size-lg-label	Label font size for large variant

Best Practices

Design Guidelines

- Provide clear options - Use descriptive labels that clearly distinguish choices
- Limit option count - Keep radio groups to 7 or fewer options for usability
- Choose appropriate orientation - Use vertical for longer lists, horizontal for 2-4 items
- Include default selection - Pre-select the most common or safe option when appropriate
- Group related options - Only use radio buttons for mutually exclusive choices
- Consider disabled states - Disable options that aren't currently available

Performance

- Optimize option rendering - Use trackBy functions for dynamic option lists
- Debounce rapid changes - Consider debouncing for real-time validation
- Minimize re-renders - Use OnPush change detection strategy
- Efficient event handling - Use event delegation for large option lists

Form Integration

- Test with forms - Ensure proper integration with your form validation
- Handle validation - Provide clear feedback for required selections
- Consider reset behavior - Define what happens when forms are reset
- Support reactive forms - Use proper ControlValueAccessor implementation